Tianqi Zhang (001056916)

Program Structures & Algorithms

Spring 2021

Assignment No. 3

- **Task**

  Step 1:

  (a) Implement height-weighted Quick Union with Path Compression. For this, you will flesh out the class UF_HWQUPC. All you have to do is to fill in the sections marked with // TO BE IMPLEMENTED ... // ...END IMPLEMENTATION.

  (b) Check that the unit tests for this class all work. You must show "green" test results in your submission (screenshot is OK).

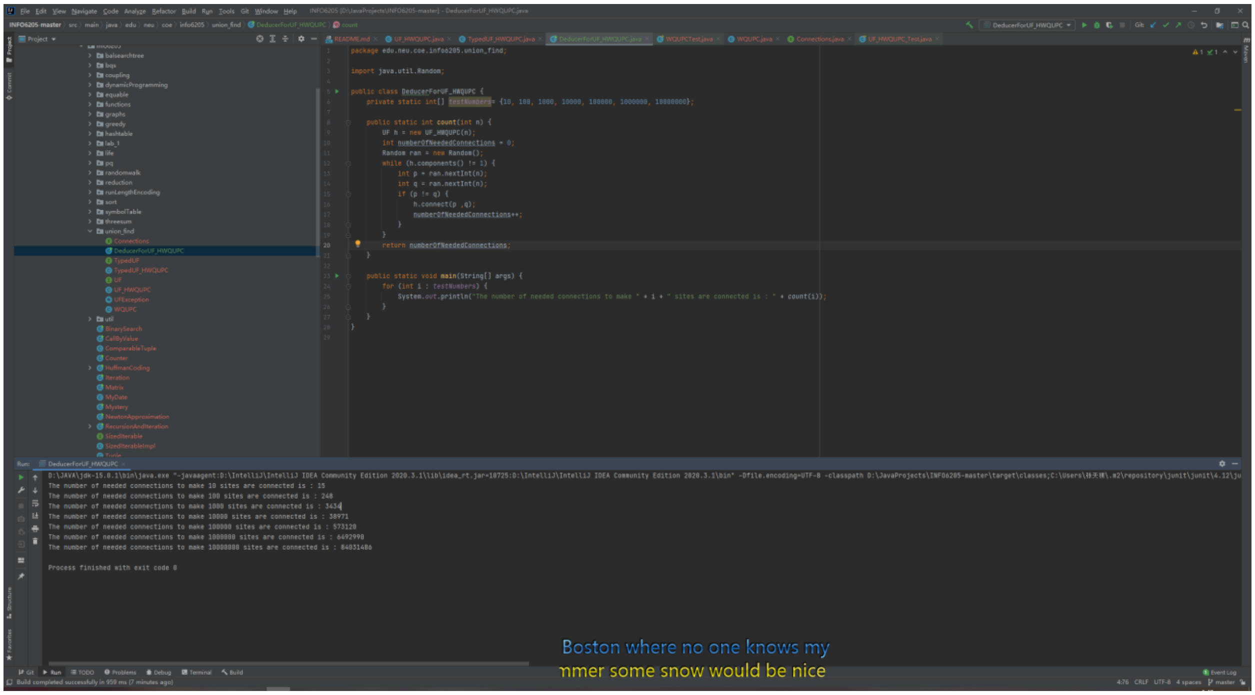
  Step 2:

  Using your implementation of UF_HWQUPC, develop a UF ("union-find") client that takes an integer value n from the command line to determine the number of "sites." Then generates random pairs of integers between 0 and n-1, calling connected() to determine if they are connected and union() if not. Loop until all sites are connected then print the number of connections generated. Package your program as a static method count() that takes n as the argument and returns the number of connections; and a main() that takes n from the command line, calls count() and prints the returned value. If you prefer, you can create a main program that doesn't require any input and runs the experiment for a fixed set of n values. Show evidence of your run(s).

Step 3:
Determine the relationship between the number of objects (n) and the number of pairs (m) generated to accomplish this (i.e. to reduce the number of components from n to 1). Justify your conclusion.

- **Output**

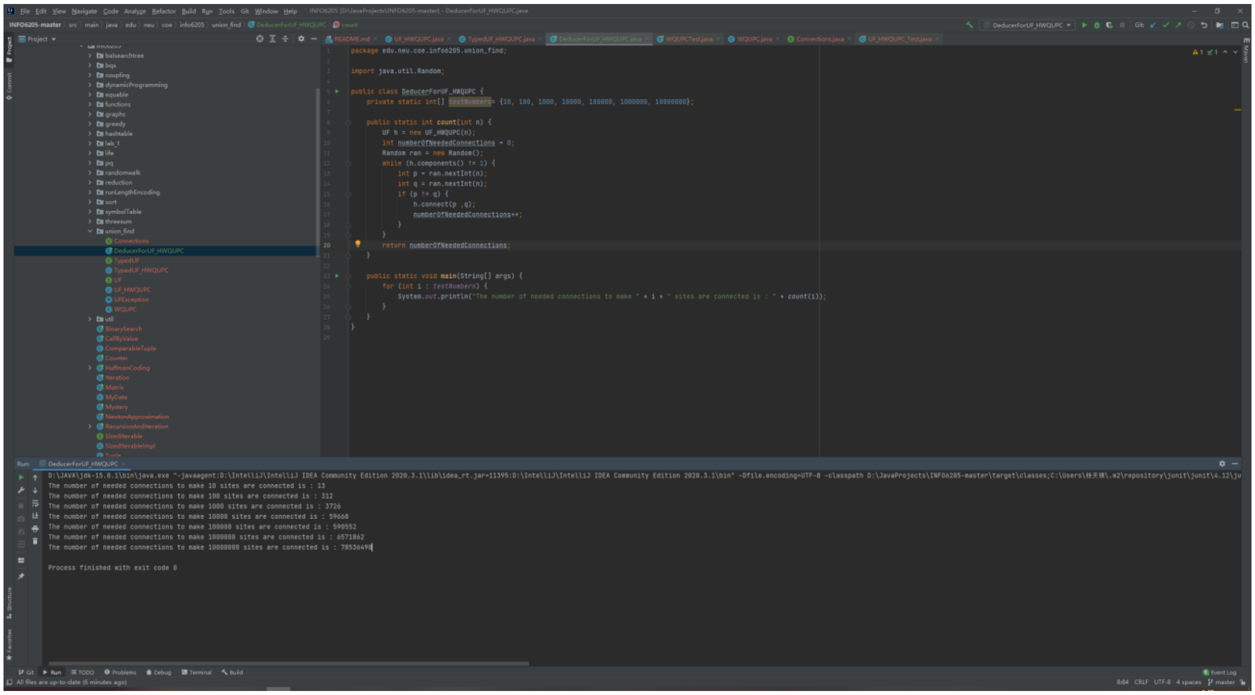
Output1:



```java
package edu.neu.coe.info6205.union_find;

import java.util.Random;

public class DeducerForUF_HWQUPC {
    private static int[] testNumbers= {10, 100, 1000, 10000, 100000, 1000000, 10000000};

    public static int count(int n) {
        UF h = new UF_HWQUPC(n);
        int numberOfNeededConnections = 0;
        Random ran = new Random();
        while (h.components() != 1) {
            int p = ran.nextInt(n);
            int q = ran.nextInt(n);
            if (p != q) {
                h.connect(p ,q);
                numberOfNeededConnections++;
            }
        }
        return numberOfNeededConnections;
    }

    public static void main(String[] args) {
        for (int i : testNumbers) {
            System.out.println("The number of needed connections to make " + i + " sites are connected is : " + count(i));
        }
    }
}
```

```
D:\JAVA\jdk-15.0.1\bin\java.exe "-javaagent:D:\IntelliJ\IntelliJ IDEA Community Edition 2020.3.1\lib\idea_rt.jar=10725:D:\IntelliJ\IntelliJ IDEA Community Edition 2020.3.1\bin" -Dfile.encoding=UTF-8 -classpath D:\JavaProjects\INFO6205-master\target\classes;C:\Users\张天祺\.m2\repository\junit\junit\4.12\ju
The number of needed connections to make 10 sites are connected is : 15
The number of needed connections to make 100 sites are connected is : 248
The number of needed connections to make 1000 sites are connected is : 3434
The number of needed connections to make 10000 sites are connected is : 38971
The number of needed connections to make 100000 sites are connected is : 573120
The number of needed connections to make 1000000 sites are connected is : 6492998
The number of needed connections to make 10000000 sites are connected is : 84031486

Process finished with exit code 0
```

Output2:



```java
package edu.neu.coe.info6205.union_find;

import java.util.Random;

public class DeducerForUF_HWQUPC {
    private static int[] testNumbers= {10, 100, 1000, 10000, 100000, 1000000, 10000000};

    public static int count(int n) {
        UF h = new UF_HWQUPC(n);
        int numberOfNeededConnections = 0;
        Random ran = new Random();
        while (h.components() != 1) {
            int p = ran.nextInt(n);
            int q = ran.nextInt(n);
            if (p != q) {
                h.connect(p ,q);
                numberOfNeededConnections++;
            }
        }
        return numberOfNeededConnections;
    }

    public static void main(String[] args) {
        for (int i : testNumbers) {
            System.out.println("The number of needed connections to make " + i + " sites are connected is : " + count(i));
        }
    }
}
```

```
D:\JAVA\jdk-15.0.1\bin\java.exe "-javaagent:D:\IntelliJ\IntelliJ IDEA Community Edition 2020.3.1\lib\idea_rt.jar=11395:D:\IntelliJ\IntelliJ IDEA Community Edition 2020.3.1\bin" -Dfile.encoding=UTF-8 -classpath D:\JavaProjects\INFO6205-master\target\classes;C:\Users\张天祺\.m2\repository\junit\junit\4.12\ju
The number of needed connections to make 10 sites are connected is : 13
The number of needed connections to make 100 sites are connected is : 312
The number of needed connections to make 1000 sites are connected is : 3726
The number of needed connections to make 10000 sites are connected is : 59668
The number of needed connections to make 100000 sites are connected is : 590552
The number of needed connections to make 1000000 sites are connected is : 6571862
The number of needed connections to make 10000000 sites are connected is : 78536498

Process finished with exit code 0
```

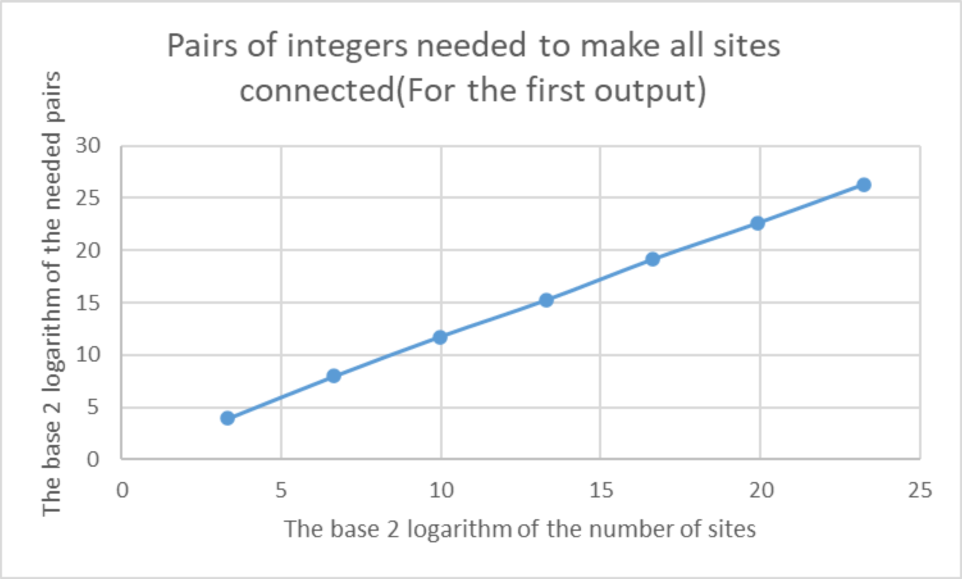
- **Relationship Conclusion:**

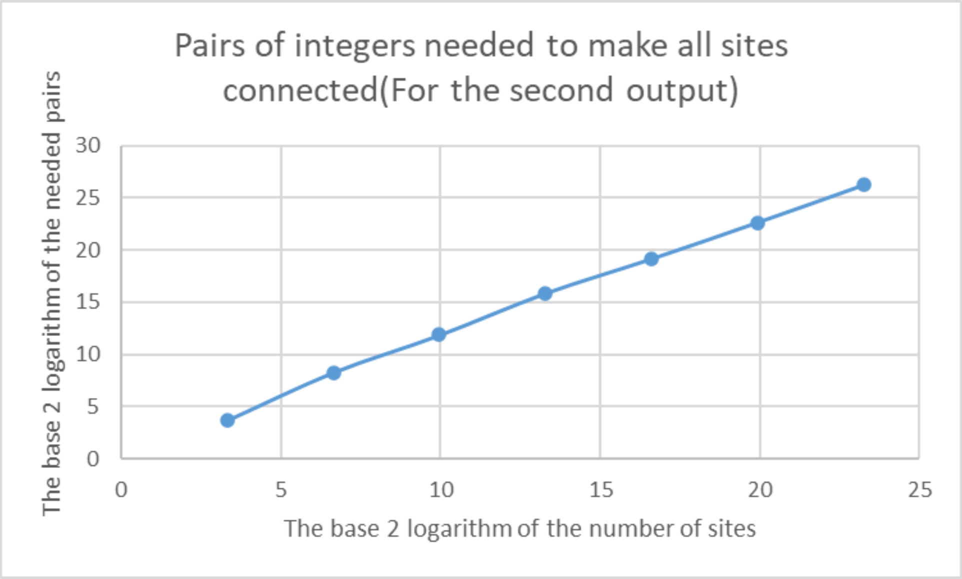
The relationship between the number of objects ***n*** and the number of pairs ***m*** generated to accomplish this task is :

$\log \boldsymbol{m} = k \log \boldsymbol{n}$(k is constant and around 1.1 or 1.2)

Evidence to support the conclusion:

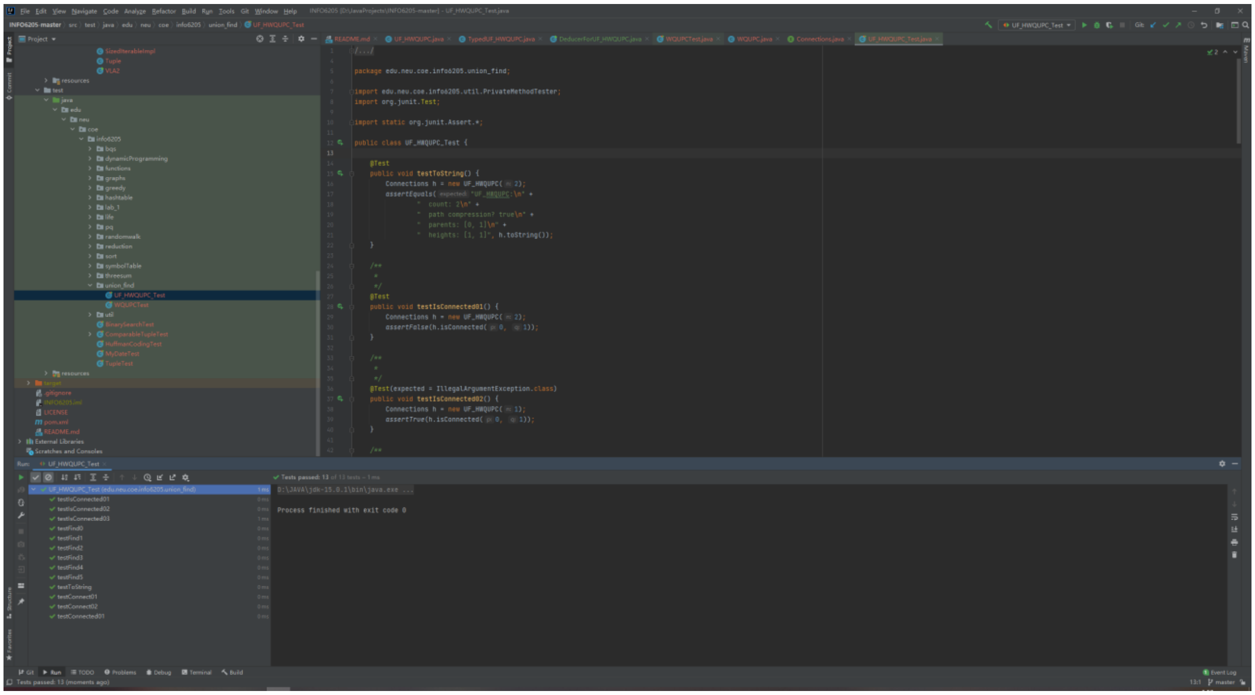
- **Graphical representation:**





- **Unit tests result:**

UF_HWQUPC_Test:



INFO6205 [D:\JavaProjects\INFO6205-master] - UF_HWQUPC_Test.java

```java
/.../

package edu.neu.coe.info6205.union_find;

import edu.neu.coe.info6205.util.PrivateMethodTester;
import org.junit.Test;

import static org.junit.Assert.*;

public class UF_HWQUPC_Test {

    @Test
    public void testToString() {
        Connections h = new UF_HWQUPC(2);
        assertEquals("UF_HWQUPC:\n" +
                "  count: 2\n" +
                "  path compression? true\n" +
                "  parents: [0, 1]\n" +
                "  heights: [1, 1]", h.toString());
    }

    /**
     *
     */
    @Test
    public void testIsConnected01() {
        Connections h = new UF_HWQUPC(2);
        assertFalse(h.isConnected(0, 1));
    }

    /**
     *
     */
    @Test(expected = IllegalArgumentException.class)
    public void testIsConnected02() {
        Connections h = new UF_HWQUPC(1);
        assertTrue(h.isConnected(0, 1));
    }

    /**
```

Run: UF_HWQUPC_Test

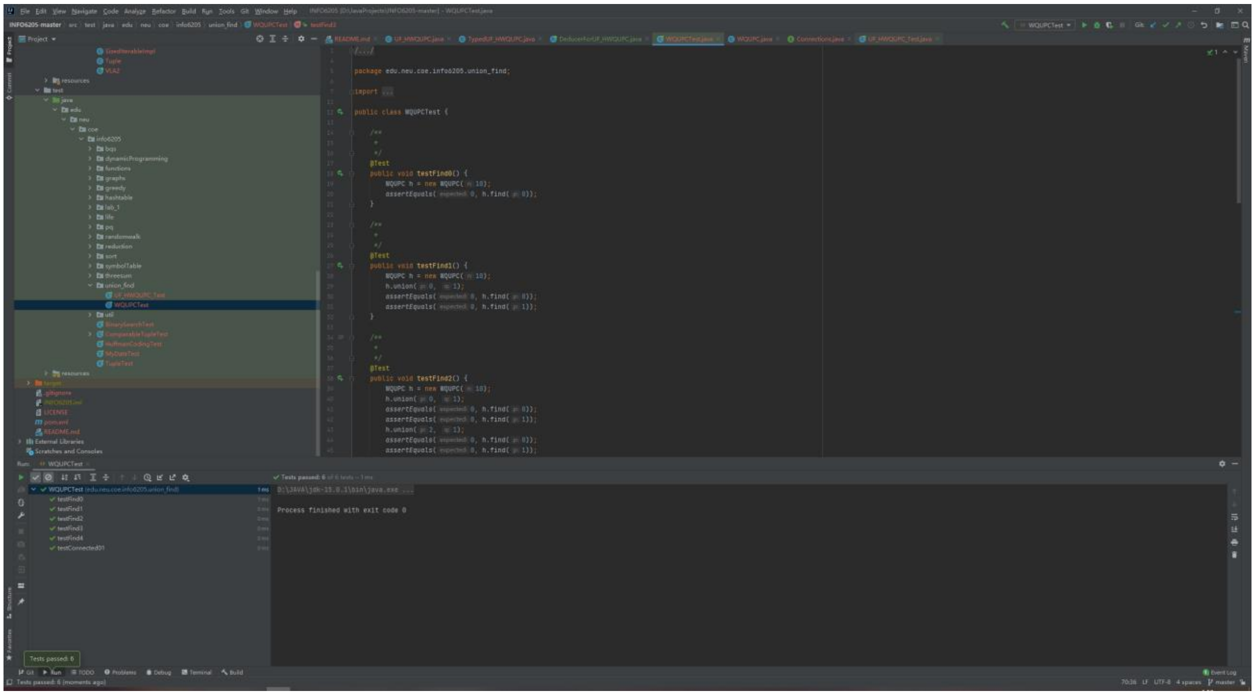
✓ Tests passed: 13 of 13 tests – 1 ms

```
✓ UF_HWQUPC_Test (edu.neu.coe.info6205.union_find)   1 ms
    ✓ testIsConnected01    0 ms
    ✓ testIsConnected02    0 ms
    ✓ testIsConnected03    1 ms
    ✓ testFind0            0 ms
    ✓ testFind1            0 ms
    ✓ testFind2            0 ms
    ✓ testFind3            0 ms
    ✓ testFind4            0 ms
    ✓ testFind5            0 ms
    ✓ testToString         0 ms
    ✓ testConnect01        0 ms
    ✓ testConnect02        0 ms
    ✓ testConnected01      0 ms
```

```
D:\JAVA\jdk-15.0.1\bin\java.exe ...

Process finished with exit code 0
```

Tests passed: 13 (moments ago)

WQUPCTest:



INFO6205 [D:\JavaProjects\INFO6205-master] - WQUPCTest.java

```java
/.../

package edu.neu.coe.info6205.union_find;

import ...

public class WQUPCTest {

    /**
     *
     */
    @Test
    public void testFind0() {
        WQUPC h = new WQUPC(10);
        assertEquals(0, h.find(0));
    }

    /**
     *
     */
    @Test
    public void testFind1() {
        WQUPC h = new WQUPC(10);
        h.union(0, 1);
        assertEquals(0, h.find(0));
        assertEquals(0, h.find(1));
    }

    /**
     *
     */
    @Test
    public void testFind2() {
        WQUPC h = new WQUPC(10);
        h.union(0, 1);
        assertEquals(0, h.find(0));
        assertEquals(0, h.find(1));
        h.union(2, 1);
        assertEquals(0, h.find(0));
        assertEquals(0, h.find(1));
```

Run: WQUPCTest

✓ Tests passed: 6 of 6 tests – 1 ms

```
✓ WQUPCTest (edu.neu.coe.info6205.union_find)   1 ms
    ✓ testFind0
    ✓ testFind1
    ✓ testFind2
    ✓ testFind3
    ✓ testFind4
    ✓ testConnected01
```

```
D:\JAVA\jdk-15.0.1\bin\java.exe ...

Process finished with exit code 0
```

Tests passed: 6

Tests passed: 6 (moments ago)